Appendix I

```
int Browser_Handler::process_request( const char * buffer, ssize_t r_bytes )
{
 m_n_bytes_received += r_bytes;

 if ( (m_n_bytes_to_receive != 0) &&
     (m_n_bytes_received > (m_n_bytes_to_receive + 2)) )
 {
  reset();
  m_n_bytes_received = r_bytes;
 }

 if ( m_o_request.bypass_parsing() == false)
 {
  IKENA_TRACE  ( DBG_IKPROXY_DUMP, ( "Browser_Handler(%x): Dumping raw
incoming data...\n", this ));
  IKENA_HEXDUMP( DBG_IKPROXY_DUMP, ( buffer, r_bytes ));

  switch ( m_o_request.inject(buffer, r_bytes) )
  {
  case IR_COMPLETE:
   if ( m_o_request.parse() == PR_ERROR)
   {
    m_response_buf.copy( INVALID_REQUEST_MSG,
INVALID_REQUEST_MSG_LEN );
    send_i();
    return CONNECTION_CLOSE;
   }

   if ( (strlen(m_o_request.get_request_host()) == CMD_INFO_VIEW_LEN) &&
       stricmp(m_o_request.get_request_host(), CMD_INFO_VIEW) == 0 )
   {
    create_info_page();
    return CONNECTION_CLOSE;
   }
   else if ( (strlen(m_o_request.get_request_host()) == CMD_PURGE_COOKIE_LEN) &&
       stricmp(m_o_request.get_request_host(), CMD_PURGE_COOKIE) == 0 )
   {
    return CONNECTION_CLOSE;
   }

   m_b_shared_session = m_o_request.is_shared_session();

   if ( m_b_shared_session == true )
   {
    m_o_request.destroy_cookies();
```

```
     m_o_request.set_cookies( m_o_cookie_jar.get_cookies(m_o_request.get_request_host(),
m_o_request.get_URL()) );
    }

    m_temp_buffer.length(0);
    m_temp_buffer.crunch();
    m_o_request.compose(m_temp_buffer);

    IKENA_TRACE  ( DBG_IKPROXY_DUMP, ( "Browser_Handler(%x): Dumping parsed
data...\n", this ));
    IKENA_HEXDUMP( DBG_IKPROXY_DUMP, ( m_temp_buffer.rd_ptr(),
m_temp_buffer.length() ));

    m_web_handler.set_request_host( m_o_request.get_request_host() );
    m_web_handler.set_request_port( m_o_request.get_request_port() );

    if ( m_web_handler.start( m_b_shared_session, m_o_request.get_host(),
m_o_request.get_URL(), m_o_request.get_port()) < 0 )
    {
     return CONNECTION_CLOSE;
    }

    m_web_handler.inject_request_data( m_temp_buffer.rd_ptr(), m_temp_buffer.length() );


    if ( m_o_request.get_body_length() > 0)
    {
     m_n_bytes_to_receive = m_o_request.get_body_length() +
m_o_request.get_unmodified_header_length();
     IKENA_TRACE( DBG_IKPROXY, ( "Browser_Handler(%x): SET
m_n_bytes_to_receive = %d...\n", this, m_n_bytes_to_receive ));

     ACE_ASSERT( (m_n_bytes_to_receive + 2) >= m_n_bytes_received );
    }
    else
    {
     m_n_bytes_to_receive = m_n_bytes_received = 0;

     IKENA_TRACE( DBG_IKPROXY, ( "Browser_Handler(%x): SET
m_n_bytes_to_receive = UNKNOWN...\n", this ));
    }

    m_web_handler.send();

    if ( m_n_bytes_received == m_n_bytes_to_receive )
    {
      reset();
     }
     break;

   case IR_ERROR:
      m_response_buf.copy( INVALID_REQUEST_MSG,
INVALID_REQUEST_MSG_LEN );
      send_i();
      return CONNECTION_CLOSE;
     break;

   case IR_INCOMPLETE:
     IKENA_TRACE( DBG_IKPROXY, ( "Browser_Handler(%x): Complete header NOT
yet given\n", this ));
     break;

   default:
     ACE_ASSERT(false);
     break;
   }
  }
  else
  {
   IKENA_TRACE( DBG_IKPROXY, ( "Browser_Handler(%x): Bypassing parse data...\n",
this ));

   if ( m_n_bytes_received > (m_n_bytes_to_receive + 2) )
   {
     ACE_ASSERT( false );
   }
   else
   {
     m_web_handler.inject_request_data( buffer, r_bytes );
   }

   m_web_handler.send();

   if ( m_n_bytes_received >= m_n_bytes_to_receive )
   {
     reset();
   }
  }

  return CONNECTION_ALIVE;
 }
```

```
inline void HTTP_Request::destroy_cookies( void )
{
 m_v_cookies.clear();
 return;
}

inline void HTTP_Request::set_cookies( const vector<HTTP_Cookie> * v_cookies )
{
 if ( v_cookies != 0 )
 {
  m_v_cookies = *(v_cookies);
 }

 return;
}
```